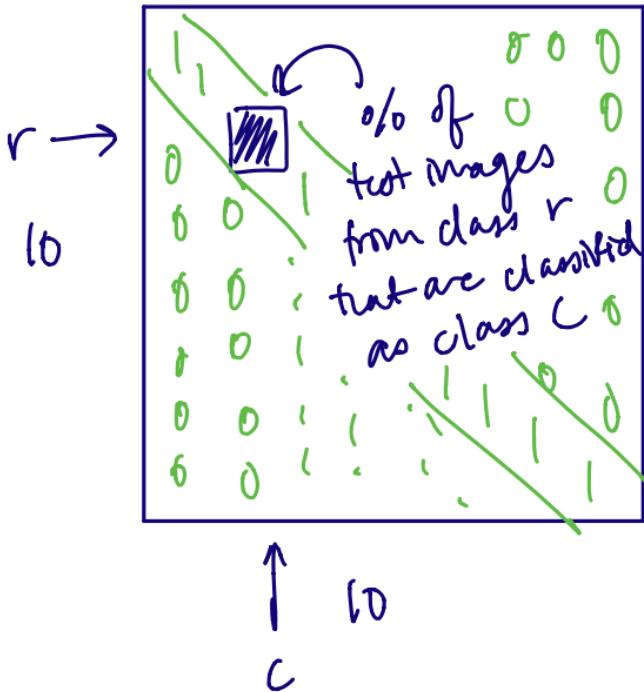


Confusion Matrix



0	1	2	3	4	5	6	7	8	9
0
1
2
3
4
5
6
7
8
9

how many test images from class 1 were defined as class 9?

Confusion Matrix [1][9] = $\frac{\# \text{ of class 1 images defined as 9}}{\# \text{ of data 1 images}}$

- > find indexes of "i" class images in testLabels (indexes of Labels)
- > go through each index, if value of results at that index == "j", increment counter (value counter)
- > populate confusionMatrix [i][j] with $\left(\frac{\text{value Counter}}{\text{indexes of Labels.size()}} \right) 100$.

Naive Bayes Outline

Tasks

- Read training data from files and generate a model
 - Save a model as a file
 - Load a model from a file
 - Classify images from a file

} Training Stage

} classifying stage

(C) ReadData class

- parse data file

① Model class

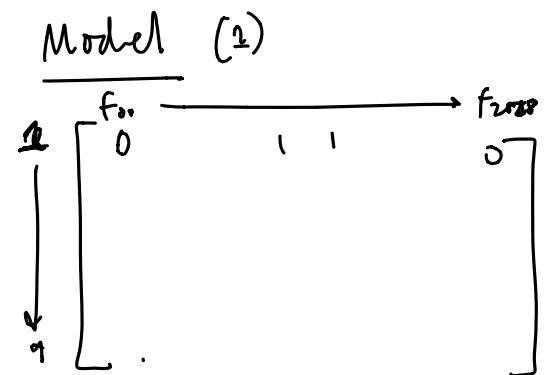
- find $P(F_{00} = 0 | \text{class} = "1")$, $P(F_{00} = 1 | \text{class} = "1")$
 $P(F_{01} = 0 | \text{class} = "1")$, $P(F_{01} = 1 | \text{class} = "1")$
 $P(F_{02} = 0 | \text{class} = "1")$, $P(F_{02} = 1 | \text{class} = "1")$
 \vdots
 $P(F_{2828} = 0 | \text{class} = "1")$, $P(F_{2828} = 1 | \text{class} = "1")$

(generate array model of probabilities for class 1)

model possibilities

> 2D Vectors (2×784) (9×784)

> 3D Vector ($28 \times 28 \times 2$)



> priors : $P(\text{class} = c) = \frac{\# \text{ of training examples where } \text{class} = c}{\# \text{ of training examples}}$

$$P(\text{class} = 1) = \frac{\# \text{ of training examples where class} = 1}{\# \text{ of training examples}}$$

$$\text{prior vector} = \left[\underbrace{0.~}_{0}, \underbrace{~}_{1}, \underbrace{~}_{2}, \underbrace{~}_{3}, \underbrace{~}_{4}, \underbrace{~}_{5}, \underbrace{~}_{6}, \underbrace{~}_{7}, \underbrace{~}_{8} \right]$$

all prior probabilities

Save model as
model instance

⑥ Read Model Class → 2D vector

- Class 1 probability =

- $P(f_{1,1} \mid \text{class 1}) = \dots$

posterior probabilities vector $\left[\underbrace{\dots}_{\text{1}}, \dots \right]$

- highest values is the correct class!

Steps : training

> start with "ZERO"

→ Using "trainiglabels", find indexes of all zero images

> using "training images", start with first pixel in each image @' indexes

- if pixel = '#' or '+', increment counter.
 - length of indexes_of_images is total # of examples where class = ONE.

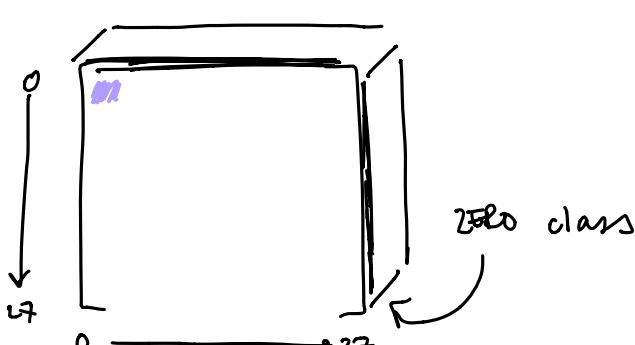
$$= P(F_{i,j} = f | \text{class} = c) = \frac{k + \text{\# of times } F_{i,j} = f \text{ when class} = c}{2k + \text{Total number of training examples where class} = c}$$

arbitrary
 ↑
 on counter

indexes_of_images.length

> populate model Array with probability found :

model Array [2580] [0] [0] = **0**



> repeat process for all pixels

> repeat process for all digits

This will result in a model with all pixels/levels filled.

* write this model to a file to save it.



Steps : classifying

* what you have : file of images, textfile with model

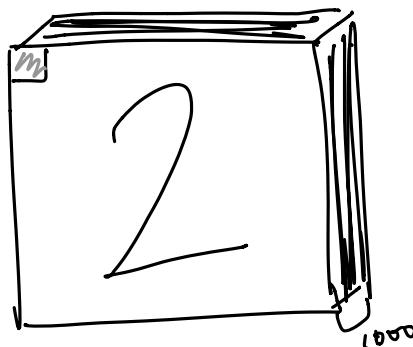
> read model.txt and populate probabilityVector

> read "testImages.txt" and populate testImagesVector

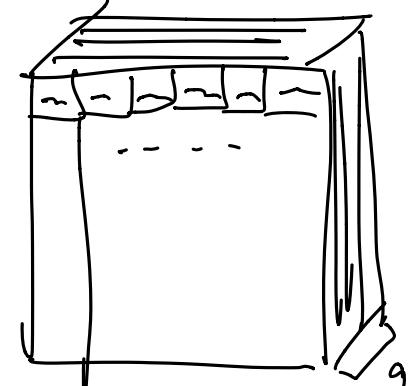
> posterior Probability Vector is float with 10 spots

> $PPV[\text{digit}] = \underbrace{\text{priorVector}[\text{digit}] + \text{probability}[\text{digit}][0][0] + \dots + \text{probability}[\text{digit}][28][28]}_{\text{posterior probability}}$

posterior probability: $\log(P(\text{class})) + \log(P(f_{1,1}|\text{class})) + \log(P(f_{1,2}|\text{class})) + \dots + \log(P(f_{28,28}|\text{class}))$



if ($m = \# || +$) \rightarrow probabilityVector[digit][m]
else \rightarrow 1 - probabilityVector[digit][m]



> start @ first pixel

> check if m/off

• if ($m = \# || +$) \rightarrow probabilityVector[digit][m]
else \rightarrow 1 - probabilityVector[digit][m]

④ Detector Class
testImagesVector
probabilityVector
posterior probabilityVector
priorVector

> start with first image in testImages

Trainer Class

- training_images [][][]
- training_labels []
- priors []
- model [][][]
- numberOfTimesDigitOccurs []

> getters for all variables (not # of times)

> set Training Images (file)

↳ put stuff from file into training_images

> set Training Labels (file)

↳ put stuff from file into trainingLabels

> set # of times digit occurs (trainingLabels [])

↳ populate numberOfTimesDigitOccurs []

> set priors (# of times Vector)

↳ populate priors

> set Model (trainingLabels, trainingImages)

↳ populate model

initialize Trainer

Detector Class

- test_images [][][]
- test_labels []
- posteriors [][]
- results [] → size = 1000
- confusion_matrix []
- model [][][] } pass into constructor
- priors []

> getters for all variables

> set Test Images (file)

↳ put stuff from file into test_images

> set Test Labels (file)

↳ put stuff from file into test_labels

> set Posteriors (testImages)

↳ populate posteriors for every page

> set Results (posteriors)

↳ calculate max of array row of posteriors, populate results array

> set Confusion Matrix (testlabels, results)

↳ compare and populate

generateResults